1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: “data exploration”, “outlier investigation”]

The goal of this project is to take the dataset made up of financial information and emails corresponding to various individuals and build a program that is able to determine who is a Person of Interest related to the Enron scandal from this information. Machine
learning was useful in processing the large dataset quickly and finding out the person of interests based on key features from the dataset.

The Enron scandal is one of the largest corporate scandals ever to take place in the US. There are 146 Enron employees in the dataset, 18 of them are POIs. There are 14 financial
features and 6 email features.

The outliers found in the dataset were namely “TOTAL” and “THE TRAVEL AGENCY IN THE PARK”, which was removed.

2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]

Selected features (feature score) -:

- salary (16.279540774)
- bonus (9.25774802884)
- bonus_salary_ratio (36.7788999845)
- shared_receipt_with_poi (25.6122723344)
- from_this_person_to_poi_ratio (7.80938837534)

The selection process was carried out using an automated feature selection function ‘SelectKBest’. No scaling was carried out because scaling isn't required for tree-based algorithms because splitting of the data is based on a threshold value.

Engineered Features:

- bonus_salary_ratio
- from_to_poi_ratio
- from_this_person_to_poi_ratio
- from_poi_to_this_person_ratio

➢ “Bonus to salary ratio” would explain high bonus and low salary or vice versa in certain employees.
➢ The ratio of “from and to poi ratio” would explain the communication the employees had with pois.
➢ The features ‘ from_this_person_to_poi_ratio’ and ‘from_poi_to_this_person_ratio’ explains how much an employee is in contact with a poc compared to other employees.

“SelectKBest” was used to find the best feature among all the features. The “k” was tuned & tested with GridSearchCV with k=5.

The reason for using feature selection was because the precision value returned by the program without feature selection was **0.24**, and after feature selection the precision value returned is **0.308.**

3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: “pick an algorithm”]

The various algorithms I tried was DecisionTree, Random Forest & Kneighbors. Out of these 3, I decided to go with Decision Tree algorithm as it had the Precision & Recall value above 0.3

	Accuracy	**Precision**	**Recall**
KNeighbors	0.84	0.30	0.14
Decision Tree	0.73	0.30	0.77
Random Forest	0.85	0.36	0.12

4. What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well? How did you tune the parameters of your particular algorithm? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: “tune the algorithm”]

Tuning is the process of optimizing the algorithm so that the best result is achieved during execution of the algorithm. Many of the ML classification algorithms have various parameters that have one or more optional settings.

Parameters for DecisionTreeClassifier:

- class_weight=[None, 'balanced'],
- criterion=['entropy'],
- max_depth=[None, 2, 3, 4,5],
- min_samples_split=[2, 3, 4, 25]

Tuning for this was done by selecting “Entropy” as the criterion , setting the max_depth & min_sample_split

The way I approach parameter selection is by using GridSearchCV to test every combination of algorithm parameter settings that it chooses, and use the algorithm to select the best one according an average of metrics returned.

5. What is validation, and what’s a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: “validation strategy”]

Validation in the context of machine learning is basically testing how well the model that one has generated on the dataset can work on data outside of that training data -- that is, how generalizable is it to the "real world". A classic validation mistake is testing the algorithm on the same data that it is trained on. A test train split of the dataset is required to capture the true performance of the algorithm

The tester.py’s Stratified Shuffle Split cross validation was used to gauge the algorithm’s performance

6. Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable

The evaluation metrics I mostly focused on were Accuracy, Precision, and Recall. The average precision for the Decision tree classifier was 0.309 and the average recall was 0.770.

	Accuracy	**Precision**	**Recall**
KNeighbors	0.84	0.30	0.14
Decision Tree	0.73	0.30	0.77
Random Forest	0.85	0.36	0.12

Precision is how often a class prediction is right when we guess that class & Recall is how often we predict the label class correctly when the class has truly occurred.

In a dataset where most people are not POIs, say 20/150, the algorithm may just classify everyone a non POI and still have high accuracy. Using Recall (true positives/(true positives+false negatives) and Precision (True Positive / (True Positive + False Positive) give us a better understanding of how the classifier performs